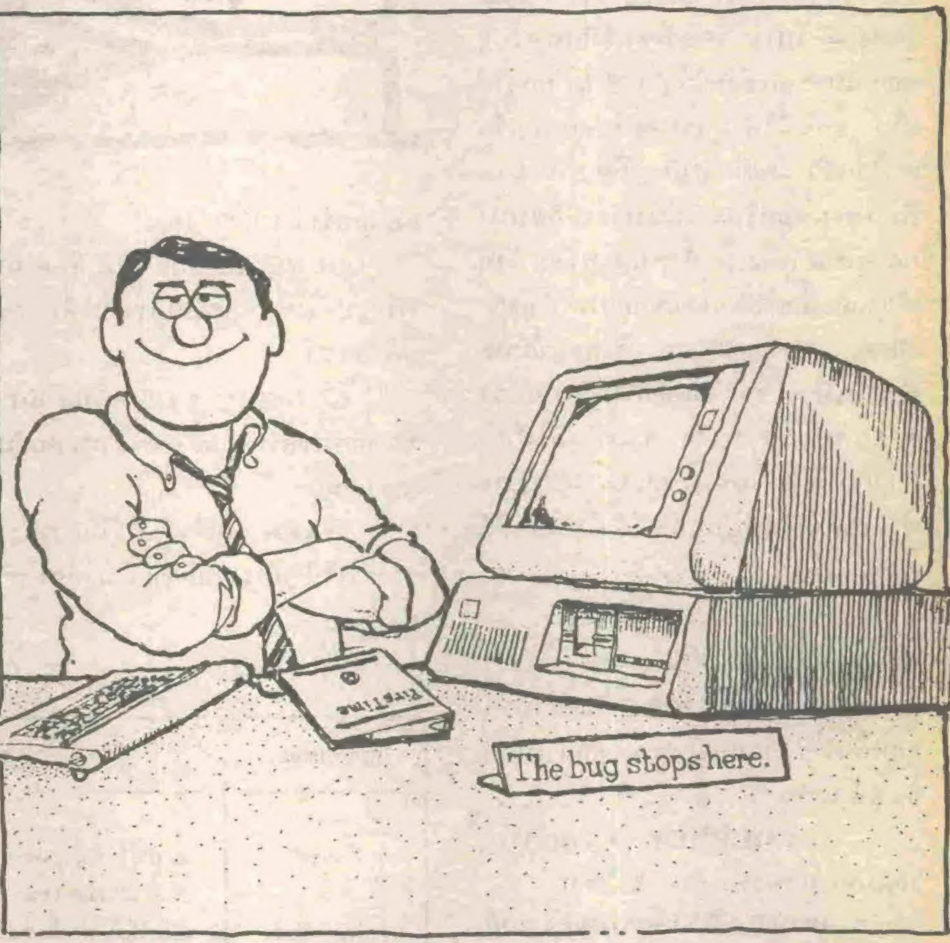
อนุชาติ ลาสนธิ

ป้องกันข้อผิดพลาด มาล็อคคีย์บอร์ดกันเถอะ

คุณคีย์ข้อมูลไว้เยอะแยะ มีมือดีทำข้อมูลอันตรธาน เรามาหาวิธีป้องกัน ด้วยการล็อคคีย์บอร์ดกันเถอะ

เมื่อครั้งที่ผู้เขียนซื้อคอมพิวเตอร์มาใหม่ ๆ ทุกคนในบ้านต่างก็ตื่นเต้นกันน่าดูเลย รวมทั้งน้องเล็ก ๆ 2 คน ที่ชอบมานั่งดูผู้เขียนนั่งคลุกอยู่กับมันนาน ๆ ก็จะให้กดคนละทีสองทีพากันดีใจใหญ่มีอยู่ครั้งหนึ่ง ผู้เขียนใช้เวิร์ดโปรเซสเซอร์ภาษาไทยตัวหนึ่งพิมพ์รายงาน ประมาณ 2 ชั่วโมง พิมพ์ได้ไปแล้วเกือบร้อยบรรทัด บังเอิญท่านบุรุษไปรษณีย์มาบีบแตรเรียก ด้วยความไม่ระวัง 2 นักโปรแกรมมั่วที่นั่งอยู่ด้านข้าง ผู้เขียนก็ออกไปรับจดหมาย พอกลับเข้ามาไม่รู้ว่าไอ้ 2 นักโปรแกรมหายตัวไปไหน เมื่อจะพิมพ์งานต่อปรากฏว่าโปรแกรมไม่รับรู้คีย์ที่กดเลย ไม่ต้องถามครับว่าใครคือมือดีสำหรับงานนี้ เป็นอันว่าที่พิมพ์มา 2 ชั่วโมงสูญเปล่าไปเพราะมีมือดีลึกลับที่ไม่ได้รับอนุญาตมาคีย์ช่วย


The bug stops here.

ก็เลยเป็นต้นเหตุให้ผู้เขียนเขียนโปรแกรมล็อคคีย์บอร์ดมาแจกให้ใช้กัน บางท่านที่มีเครื่องเป็นของตัวเองและไม่ค่อยมีคนไปยุ่งด้วยคงไม่มีปัญหา แต่บางท่านใช้เครื่องที่สถาบัน คนที่ใช้เครื่องก็ย่อมจะมีมากเป็นธรรมดา โปรแกรมนี้ช่วยให้คุณสบายใจได้ว่าในขณะที่คุณละทิ้งเครื่องชั่วครู่ นักโปรแกรมมั่วที่อยู่แถว ๆ นั้นจะยุ่งกับงานของเราไม่ได้

คุณสมบัติของโปรแกรม

คุณสมบัติของโปรแกรมล็อคคีย์บอร์ด เมื่อขณะนั้นโปรแกรมอยู่ในโหมดการล็อค ทุกครั้งที่มีการกดคีย์จะไม่มีผลต่อการทำงานขณะนั้น จนกว่าจะกดคีย์ที่ใช้ถอนการล็อค

หลักการ

แน่นอนครับ โปรแกรมที่จะทำงานลักษณะนี้ได้จะต้องเป็นโปรแกรมที่ฝังในหน่วยความจำ (resident) โปรแกรมก็จะคอยตรวจสอบว่า คีย์ที่กดเป็นคีย์ที่ใช้ล็อคคีย์บอร์ดรึเปล่า ถ้าใช่ก็จะทำการล็อคคีย์บอร์ด ทำให้คีย์ที่กดตามมาไม่มีผลอะไร และเมื่ออยู่ในโหมดการล็อคก็จะคอยตรวจสอบการกดคีย์ที่ใช้ถอนการ

ล็อค ถ้าใช่ก็จะทำให้คีย์บอร์ดและการทำงานทุกอย่างกลับคืนสู่ภาวะปกติ

การทำงาน

เมื่อ RUN โปรแกรม LOCKKEY.COM โปรแกรมจะฝังตัวในหน่วยความจำ เมื่อกดคีย์ CTRL + F8 (scan code = 101) เคอร์เซอร์ที่แสดงอยู่ขณะนั้นจะถูกซ่อนไว้ (hidden) มองไม่เห็น ตอนนี้โปรแกรมจะล็อคคีย์บอร์ดเอาไว้แล้ว เมื่อมีการกดคีย์ใด ๆ จอภาพที่แสดงผลอยู่ในขณะนั้นจะเปลี่ยนมาแสดงเป็น reverse ทั้งหมด พร้อมทั้งมีเสียงและที่กลางจอจะมีข้อความ 'DANGER.....DANGER!!' (ถึงตอนนี้มือดีที่เข้ามามั่วอาจจะวิ่งแล้วก็ได้) เมื่อต้องการกลับสู่การทำงานตามปกติให้กด CTRL + F9 (scan code = 102) การทำงานก็จะเหมือนตอนที่ยังไม่ล็อคคีย์ทุกอย่างครับ !

อธิบายโปรแกรม

- **NUMBER_CYCLES** จำนวนลูกคลื่นที่ส่งออกลำโพง หรือความยาวของเสียง
- **FREQUENCY** ความถี่หรือระดับของเสียงที่ส่งออกลำโพง
- **PORT_B** แอดเดรสของ port ที่ต่อกับลำโพง
- **KEYLOCK** scan code ที่ใช้ในการล็อคคีย์ ตามโปรแกรมมีค่า 101 = Ctrl + F8
- **UNLOCK** scan code ที่ใช้ในการถอนล็อคคีย์ ตามโปรแกรมมีค่า 102 = Ctrl + F9
- **VIDEO_BUFFER** แอดเดรสหน่วยความจำของจอภาพ ตามโปรแกรมมีค่า 0B000H เป็นจอโมโนโครม
- **OLDVECT** เก็บอินเตอร์รัพต์เวคเตอร์ของ INT 16H

CURSOR_SHAPE เก็บขนาดหรือรูปร่างของเคอร์เซอร์ก่อนที่จะซ่อนเคอร์เซอร์

CURSOR_POS เก็บตำแหน่งของเคอร์เซอร์ทั้งแถว (row) และคอลัมน์ (column)

MARK เก็บค่าเพื่อใช้ในการตรวจสอบว่าโปรแกรมเคย run มาก่อนหรือยัง

หน้าที่ของโปรแกรมย่อยต่าง ๆ ในโปรแกรมแสดงดังตารางที่ 1

ก่อนที่จะใช้งาน

เนื่องจากโปรแกรมที่ให้มายังเป็นภาษาแอสเซมบลีอยู่ จะต้องผ่านการแปลงให้เป็นภาษาเครื่องก่อนถึงจะ run ได้โดยขั้นตอนดังรูปที่ 1 ส่วนตัวโปรแกรมนั้นแสดงดังรูปที่ 2

โปรแกรมย่อย	หน้าที่
WAIT_ESC	วนรอบรับตัวอักษรจากคีย์บอร์ด จนกระทั่งกดคีย์ที่ใช้ถอนการล็อค (unlock) จึงจะออกจากการวนรอบ
CLEAR_BUFF	เคลียร์คีย์บอร์ดบัฟเฟอร์
SAVE_SCREEN	เก็บจอภาพที่แสดงผลอยู่ที่ PAGE 0 ขณะนั้น ไปไว้ที่ PAGE 1
REST_SCREEN	คืนจอภาพที่เก็บไว้ให้ PAGE 0
HID_CURSOR	ซ่อนเคอร์เซอร์ไม่ให้มองเห็น
SAVE_CURSOR	เก็บตำแหน่งและขนาดของเคอร์เซอร์ไว้ในตัวแปร CURSOR_SHAPE, POS
REST_SHAPE	คืนขนาดของเคอร์เซอร์
REST_POS	คืนตำแหน่งของเคอร์เซอร์
SOUND	ผลิตเสียง ความยาวของเสียงกำหนดในค่า NUMBER_CYCLE ระดับของเสียงกำหนดใน FREQUENCY
DANGER	แสดงตัวอักษร reverse (ascii 219) เต็มหน้าจอ พร้อมทั้งข้อความ DANGER.....DANGER!!

ตารางที่ 1 หน้าที่ของโปรแกรมย่อยต่าง ๆ

รูปที่ 1 ขั้นตอนการคอมพายล์โปรแกรม LOCKKEY.ASM

```
A>MASM LOCKKEY;
Microsoft (R) Macro Assembler Version 5.00
Copyright (R) Microsoft Corp 1981-1985, 1987.  All rights reserved.

LOCKKEY.ASM(3150: warning A4057: Illegal size for operand
LOCKKEY.ASM(316): warning A4057: Illegal size for operand

  51548 + 279460 Bytes symbol space free

    2 Warning Errors
    0 Severe  Errors

A>LINK LOCKKEY;

Microsoft (R) Overlay Linker   Version 3.60
Copyright (R) Microsoft Corp 1983-1987.  All rights reserved.

LINK : warning L4021: no stack segment

A>EXE2BIN LOCKKEY.EXE LOCKKEY.COM
```

สรุป

ถ้าโปรแกรมคีย์ถูกต้องทั้งหมด การแปลงเป็นภาษาเครื่องจะมีการเตือน 2 บรรทัด (2 warning) ไม่มีการผิดพลาด (0 error) ถึงตอนนี้ก็ได้โปรแกรม LOCKKEY.COM แล้วพร้อมที่จะ run ได้ครับ!!

รูปที่ 2 โปรแกรม LOCKKEY.ASM

```
                LOCKKEY.ASM       VER 1.0
                    for...... IBM PC/XT/AT

                    by.________________
--------------------------------------------------------

MBER_CYCLES  EQU  300               ;length of sound
EQUENCY      EQU  500               ;level of sound
RT_B         EQU  61H               ;address port B

YLOCK        EQU  101               ;scan code for lock key
LOCK         EQU  102               ;scan code for unlock

DEO_BUFFER   EQU  0B000H            ;address of video buffer
..........................................................

EG           SEGMENT
             ASSUME CS:CSEG,DS:CSEG
             ORG 100H
```

```
START:
             JMP INSTALL

;----------------------------------------------------------
OLDVECT       DD 00000000              ;save interrupt vector

CURSOR_SHAPE  DW 0000                  ;save cursor shape
CURSOR_POS    DW 0000                  ;save cursor position

DANGER_SHOW   DB 10 DUP(32)
              DB 'DANGER.....DANGER.!!'
              DB 10 DUP(32)
              DB '$'

MARK          DW 0116H                 ;mark for check program already
;---------------------------------------------------------

LOCK_KEY     PROC FAR

              Assume Cs:Cseg,Ds:Cseg
```

→

รูปที่ 2 (ต่อ)

```
            sti                     ;reenable interrupts
            or ah,ah                ;check function 0
            jz chk_func0            ;

            assume ds:nothing       ;if not function 0 then
            jmp oldvect             ;go back
;.........................................................
CHK_FUNC0:
            push bx                 ;
            push cx                 ;
            push dx                 ;
                                    ;
            push di                 ;
            push si                 ;
                                    ;
            push es                 ;
            push ds                 ;save altered registers

            push cs
            pop  ds
;...........................................................

            pushf                   ;
            call oldvect            ;get a character

            push ax                 ;save scan code and ascii code

            cmp ah,keylock          ;check scan code is lockkey
            jnz exit                ;if not equal jump exit

            call lock1              ;lock keyoard
;............................................................
EXIT:
            pop  ax                 ;restore scan code and ascii code

            pop  ds                 ;
            pop  es                 ;
                                    ;
```

```
             pop  si              ;
             pop  di              ;
                                  ;
             pop  dx              ;
             pop  cx              ;
             pop  bx              ;restore registers
;...........................................................

            IRET
LOCK_KEY    ENDP
;___________________________________________________________

LOCK1       PROC NEAR

             call save_screen     ;save screen
             call save_cursor     ;save cursor shape and position

             call hid_cursor      ;hidden cursor

             call wait_esc        ;waitig until press unlock key

             call rest_screen     ;restore screen
             call rest_shape      ;restore cursor shape
             call rest_pos        ;restore position

            RET
LOCK1       ENDP
;___________________________________________________________

WAIT ESC    PROC NEAR

REPEAT WAIT:
             call clear_buff      ;clear keyboard buffer

             pushf                ;
             call oldvect         ;get a character
            cmp ah,unlock           ;check unlock key
            jz exit_wait            ;if scan code is unlock key jump to
                                    ;exit_wait

            call danger             ;echo inverse all screen
            call sound              ;sound beep!
            jmp repeat_wait         ;go to repeat_wait
EXIT_WAIT:
            RET
WAIT_ESC    ENDP
;----------------------------------------

CLEAR_BUFF  PROC NEAR

            cli                     ;disable interrupts
            mov ax,0000             ;make AX = 0000
            mov es,ax               ;set ES to bottom memory

            mov al,es:[41aH]        ;move head point to AL register
            mov es:[41cH],al        ;place in tail pointer
            sti                     ;reenable interrupts

            RET
CLEAR_BUFF  ENDP
;----------------------------------------

SET_REG     PROC NEAR

            mov ax,video_buffer     ;AX = address of video buffer
            mov es,ax               ;make ES point to video buffer

            mov si,0000             ;SI point to offset 0000
            mov cx,4000             ;CX = number loop

            RET
```

```
SET_REG     ENDP
;----------------------------------------

SAVE_SCREEN PROC NEAR

            call set_reg            ;set ES,SI and CX registers
LOOP_SAVE:
            mov al,es:[si]          ;get character and attribute from page0
            mov es:[si+1000H],al    ;move character and attribute to page1
            inc si                  ;increment pointer
            loop loop_save          ;until CX=0000

            RET
SAVE_SCREEN ENDP
;----------------------------------------

REST_SCREEN PROC NEAR

            call set_reg
LOOP_REST:
            mov al,es:[si+1000H]    ;get character and attribute from page1
            mov es:[si],al          ;move character and attribute to page0
            inc si                  ;increment pointer
            loop loop_rest          ;until cx=0000

            RET
REST_SCREEN ENDP
;----------------------------------------

HID_CURSOR  PROC NEAR

            mov ah,01               ;function 01(set cursor shape)
            mov bh,00               ;page 0
            mov ch,0EH              ;CH=start
            mov cl,01               ;CL=end   (hidden cursor)
            int 10H                 ;set cursor shape

            RET
```

→

รูปที่ 2 (ต่อ)

```
HID_CURSOR    ENDP
;------------------------------------------------

SAVE_CURSOR   PROC NEAR

              mov ah,03                 ;function 03(get cursor shape,position)
              mov bh,00                 ;page 0
              int 10H                   ;get cursor shape,position

              mov cursor_shape,cx       ;save cursor shape
              mov cursor_pos,dx         ;save cursor position

              RET
SAVE_CURSOR   ENDP
;------------------------------------------------

REST_SHAPE    PROC NEAR

              mov ah,01                 ;function 01(set cursor shape)
              mov bh,00                 ;page 0
              mov cx,cursor_shape       ;CH=start, CL=end
              int 10H                   ;set cursor shape

              RET
REST_SHAPE    ENDP
;------------------------------------------------

REST_POS      PROC NEAR

              mov ah,02                 ;function 02(set cursor position)
              mov bh,00                 ;page 0
              mov dx,cursor_pos         ;DH=row, DL=column
              int 10H                   ;set cursor position

              RET
REST_POS      ENDP
;------------------------------------------------
```

```
SOUND         PROC NEAR

              cli                       ;disable interrupts
              mov dx,number_cycles      ;DX counts the length of the tone
              in al,port_B              ;get port B
              and al,0FEH               ;disconnect speaker from timer chip
NEXT_CYCLE:
              or al,02                  ;turn on speaker
              out port_B,al             ;send the command to port B
              mov cx,frequency          ;move the delay for 1/2 cycle to CX
FIRST_HALF:
              loop first_half           ;make delay while speaker is on
              and al,0FDH               ;turn off speaker
              out port_B,al             ;send the command to port B
              mov cx,frequency          ;move the deley for 2nd half of cycle
SECOND_HALF:
              loop second_half          ;make delay while speaker is off
              dec dx                    ;substract 1 from the number of cycles
              jnz next_cycle            ;if 0,then duration is exhausted
              sti                       ;reenable interrupts

              RET
SOUND         ENDP
;------------------------------------------------

DANGER        PROC NEAR

              call set_reg              ;set ES,SI and CX registers

              mov cx,2000               ;make CX = 2000
              mov al,219                ;AL = ascii 219
              mov ah,0FH                ;AH = highdisplay attribute
REPEAT_DGR:
              mov es:[si],al            ;move AL(character) to video buffer
              inc si                    ;increamant SI point to address of
                                        ;attribute
```

→

```
            mov es:[si],ah           ;move AH(attibute) to video buffer
            inc si                   ;increment SI point to address of
                                     ;character
            loop repeat_dgr          ;go back  repeat_dgr until CX=0000

            mov di,offset danger_show ;DI point to string danger_show
            mov si,1800              ;SI point to center of screen
            mov ah,07H               ;AL is lowdisplay attribute
SHOW_DGR:
            mov al,[di]              ;get a character from DI
            cmp al,'$'               ;check end of string
            jz exit_show             ;if AL is '$' jump to exit_show

            mov es:[si],al           ;else/ move AL to video buffer
            inc si                   ;increment SI point to address of
                                     ;attribute
            mov es:[si],ah           ;move attribute to video buffer
            inc si                   ;increment SI point to address of
                                     ;character
            inc di                   ;increment DI point to next cheracter
            jmp show_dgr             ;go to show_dgr
EXIT_SHOW:
           RET
DANGER     ENDP
;______________________________________________

INSTALL:
            push cs                  ;
            pop  ds                  ;DS = CS

            mov ah,35H               ;function 35H(get the vector)
            mov al,16H               ;
            int 21H                  ;get the vector INT 16H
                                     ;ES = CS,BX = offset

            mov ax,[bx-2]            ;AX = [BX-2]
            cmp ax,mark              ;compare AX with mark
            jnz no_ready             ;if equal then jump to no_ready
```

```
;......................................................................
            mov ah,09H              ;function 09H
            mov dx,offset ready     ;dx point to string(ready)
            int 21H                 ;echo '... LOCKKEY.COM   alrady'

            mov ah,4CH              ;functio 4CH
            int 21H                 ;exit to DOS
;......................................................................
NO_READY:
            mov oldvect,bx          ;store offset of INT 16H to oldvect
            mov oldvect[2],es       ;store CS of INT 16H to oldvect[2]

            mov ah,09H              ;function 09H
            mov dx,offset show      ;dx point to string(show)
            int 21H                 ;echo string

            mov ah,25H              ;function 25H
            mov al,16H              ;
            mov dx,offset lock_key  ;DX point to procedure lock_key
            int 21H                 ;set interrupt vector to lock_key

            mov dx,offset install   ;set offset of resident routine
            int 27H                 ;quit,and routine stays resident

;______________________________________________________________
;_____________________ DATA AREA ______________________________

SHOW        DB '.......... LOCKKEY.COM  VER 1.0 ',10,13
            DB '       BY.______________ ',10,13
            DB '$'

READY:      DB '....... LOCKKEY.COM   already . ',7,10,13
            DB '$'
;______________________________________________________________

CSEG        ENDS .
            END START
;______________________________________________________________
```